REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-12-2003	Final Report	16-01-2003 - 31-12-2003

4. TITLE AND SUBTITLE

On-demand Interactive Simulation-centered Training
For Small Unit Tactics

5a. CONTRACT NUMBER
N61339-03-C-0054

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Munro, Allen
Johnson, Mark C.
Pizzini, Quentin A.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California
Behavioral Technology Laboratories
250 N. Harbor Dr., Suite 309
Redondo Beach, CA 9077

8. PERFORMING ORGANIZATION REPORT NUMBER
Technical Report No. 124

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army RDECOM, STTC
12423 Research Parkway
Orlando, FL 32826-3276

NAVAIR Orlando TSD
12350 Research Parkway Code 25353
Orlando, FL 32826-3275

10. SPONSOR/MONITOR'S ACRONYM(S)
RDECOM-STTC
NAVAIR-ORLANDO TSD

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release: Distribution is Unlimited

13. SUPPLEMENTARY NOTES

20031217 027

14. ABSTRACT

Training on small unit infantry tactics in both the context of present-day infantry operations and in Objective Force Warrior contexts may benefit from the use of interactive graphics with behavioral complexity during training. The VIVIDS and iRides tools provide technologies for developing and delivering such training materials for delivery on Linux and Windows systems, respectively. Several tactics training topics were investigated in this research project. A prototype trainer on forward observer placement was developed using iRides. This trainer and others developed with these tools, can be delivered on small, wearable computer systems with modest graphic capabilities.

15. SUBJECT TERMS
Simulation training, Distance learning, Advanced Distributed Learning, ADL, Authoring, Training architecture, Authored training, Authored simulation, Embedded Training for Warfighters

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Donna Darling
U	U	U		29	19b. TELEPHONE NUMBER *(Include area code)* (310) 379-0844

Final Report:
On-demand Interactive Simulation-centered Training For Small Unit Tactics

Allen Munro
Mark C. Johnson
Quentin A. Pizzini

Behavioral Technology Laboratories
University of Southern California
250 N. Harbor Drive, Suite 309
Redondo Beach, CA 90277

Developed under funding by:
US ARMY PEO STRI

Under Contract No. N61339-03-C-0054

1 December 2003

USC
UNIVERSITY
OF SOUTHERN
CALIFORNIA

Final Report: On-demand Interactive Simulation-centered Training For Small Unit Tactics

PEO STRI Award No. N61339-03-C-0054

Allen Munro
Mark C. Johnson
Quentin A. Pizzini

Behavioral Technology Lab
University of Southern California
(mailto: munro@usc.edu)

Abstract

Training on small unit infantry tactics in both the context of present-day infantry operations and in Objective Force Warrior contexts may benefit from the use of interactive graphics with behavioral complexity during training. The VIVIDS and iRides tools provide technologies for developing and delivering such training materials for delivery on Linux and Windows systems, respectively. Several tactics training topics were investigated in this research project. A prototype trainer on forward observer placement was developed using iRides. This trainer and others developed with these tools, can be delivered on small, wearable computer systems with modest graphic capabilities.

Requirements and Goals

The emergence of the Objective Force Warrior (OFW) presents a set of new training requirements and new opportunities for delivering training. Each soldier equipped with an OFW system has a lightweight computer and color display. Many communications, information acquisition, data storage, and other resources will be available to OFW-equipped soldiers. At the same time, this complex system imposes its own learning requirements on the soldier. Even if soldiers are initially well trained in all the capabilities and requirements of their new systems, they will need to review these functions to reinforce that learning, and to ensure that it is immediately available for application, when required.

Many types of training benefit from interactive graphic environments. The presence of a behaviorally realistic representation of a system, subsystem, or theater of operations can provide learning opportunities that are not possible using less interactive environments,

such as text, static images, or video sequences alone. In many cases, behavioral realism is much more important than visual fidelity, because soldiers understand that depicted objects and environments may have more complex textures and shadowing, and that colors and perspectives may be somewhat different from what is depicted on screen. What the soldier needs to know is how systems behave, and how actions will determine effects in a wide variety of contexts.

In an earlier project (Munro, Pizzini, Johnson, Walker, Surmon, Dumanoir, & Garrity, 2002) dealing with on-demand interactive simulation-centered training, a small scale development process conducted with VIVIDS and iRides, authoring tools developed at Behavioral Technology Labs, showed that these tools could be used to develop training relevant to OFW systems training. Two prototype training modules for Land Warrior training were developed using these systems. One taught soldiers how to connect the numerous components of the Land Warrior I system. The second provided an interactive training environment for learning about the Land Warrior global positioning system (GPS).

In the current project, the goal was to illustrate that the same tools (VIVIDS and iRides) could be used to develop training on topics in soldier tactics, as well. The OFW wearable computer makes it possible to deliver tactical training in a time frame that may be relevant to an upcoming operation. For example, as a unit is being transported to a field of operations, the soldiers could work through the training modules that review the tactics that are planned for the operation.

Approach

VIVIDS (Munro, 1994; Pizzini, Munro, & Towne, 1996; Munro, Johnson, Pizzini, Surmon, Towne, & Wogulis, 1997; Munro & Pizzini, 1998) is an authoring and delivery system for training that makes use of behaviorally complex interactive graphics systems. Authors can draw objects, write rules that determine their interactive behavior, and write lessons that are delivered in the context of such simulations. A universal simulation engine is responsible for interpreting and interactively delivering the authored behavior. This approach supports the productive development and maintenance of simulations for training. More importantly, however, the universal simulation component has a set of services that it can offer to the instruction delivery component of the training system. This means that simulation developers do not need to re-implement such instructional services every time a new training module is developed.

The sorts of services that a simulation can offer to an instructional component include the following:

Open a simulation
Stop simulating
Start simulating
Ignore student actions
Resume responding to student actions

Carry out actions
Highlight object
Change a value
Register interest in an expression
Report expression value changes for registered expressions
... and many others

By providing such a range of services, a simulation makes it possible for a pedagogical control mechanism to observe important events in the simulation, and it gives that training component the ability to utilize the simulation for instructional purposes. For example, a tutorial can demonstrate how a procedure should be carried out in the context of the simulation, thanks to the fact that the simulator can stop responding to student actions and can instead respond to 'actions' taken by the instructional control component. Similarly, because the instructional control component can register an interest in certain conditions, it can, in effect, direct the simulator to tell it when a student makes a certain type of mistake or responds particularly well to a simulated situation. Upon being informed by the simulator, the tutorial component can engage in an appropriate instructional intervention.

VIVIDS is a Linux-based application written in C++. It can take two forms: an authoring version, and a student version, which does not allow the user to edit simulations or training specification. VIVIDS can export its simulation specifications and its lesson specifications in formats that can be utilized by iRides. iRides (Munro, 1999; Munro, Surmon, Johnson, Pizzini, & Walker 1999) is a Java application that that can be delivered over the Web to Windows, Macintosh, or Linux computers. In addition, iRides can deliver interactive graphical environments and training as a Java application on these platforms, without utilizing the Web.

The planned approach for this work was to use VIVIDS and iRides to develop and deliver new tactics-related training materials. This new training demonstration was to be relevant to objective force warrior, and, at the same time, relevant to current Army training needs.

Work Performed

Selection of training subject matter

A variety of possible tactics training topics were considered, including maneuvering and positioning, ambush tactics, calling for indirect fire, carrying out a commander's intent, and, finally, forward observer placement. We sought to find a topic area with both current infantry tactics application and objective force warrior application.

Maneuvering. Maneuvering tactics were found to cover a great many topics, including the arrangement of the soldiers during movements, the placement of the commanding officer in the field, the utilization of terrain (and micro-terrain) for cover during movement, fallback plans for movement, placement of entrenched positions with respect

to terrain, and considerations related to lines of fire. Because Objective Force Warrior maneuvering tactics are likely to vary significantly from current small unit tactics, we decided to postpone an effort in this area until additional OFW doctrine becomes available.

Ambush placement and tactics. This topic area is a rich one, but many of the tactical actions depend on the precise armament and other equipment available to the soldiers. It was therefore determined that this was not an ideal first candidate for training development, in advance of receiving detailed information about Objective Force Warrior systems.

Call for fire. Substantial efforts were expended on training how to call for fire. A brief summary of the task is presented in Appendix 3. The task is one with many features that are somewhat sensitive to particular contexts. Again, because the precise approach to call for fire in the objective force warrior context is not yet well understood, this complete topic did not appear to be ideal for a first application. However, an aspect of the call for fire task that is universal and important, and that is not fully addressed in current training, was brought to our attention, as described in the following section.

A prototype trainer on placement of observation posts

We planned to find an unmet tactics training need that could be addressed effectively using the iRides technology, and that has application in both current Army tactics training and the objective force warrior (OFW) context. LTC James G. Riley, Chief of Tactics at the Army Infantry Officers School, Ft. Benning, proposed a tractable subject for training. Micro-terrain features play a significant role in the appropriateness of a particular site as an observation post, but this role is often not understood. The commanding officer who assigns a forward observer a post may not have relevant micro-terrain information, or may have overlooked the impact of micro-terrain on a particular forward observer task. For this reason, it is important that someone assigned the Forward Observer task be aware of the commander's intent in making a forward observer assignment. Such an awareness supports carrying out the intent of a forward observer order when literal execution of the order is not possible

A training demo has been prepared that proceeds in three phases. First, it introduces the training environment in the context of a forward observer assignment. It walks the trainee through the use of the elements of the graphical user interface. Then it surrenders control to the student, so that he or she can complete the exercise. It offers hints and instruction until the student correctly carries out the procedure. In the third phase, it begins a new forward observer assignment, and offers the student the opportunity to solve it.

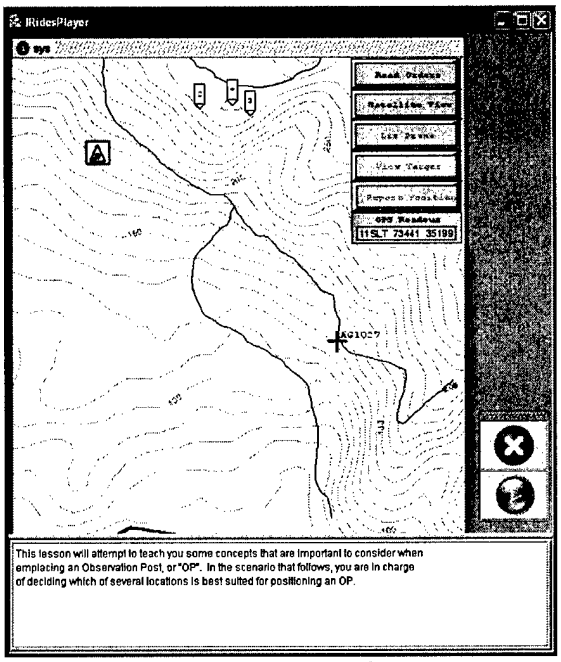

Figure 1. Introducing the GUI

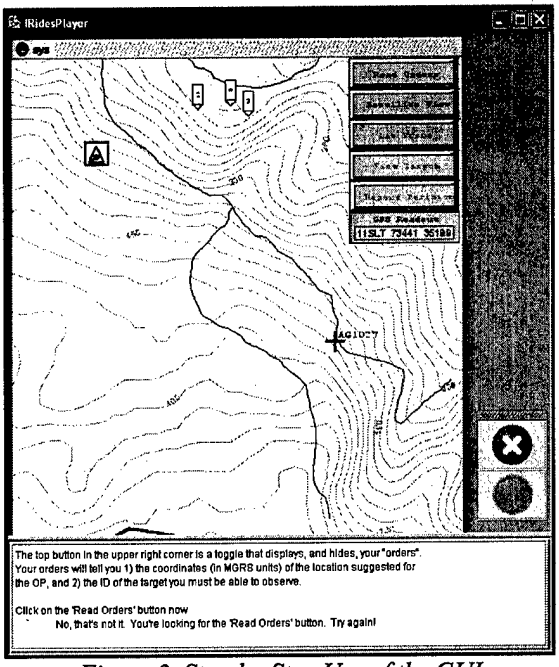

Figure 2. Step-by-Step Use of the GUI

In Figure 1, we see a simulation environment with an animated forward observer symbol and symbols that represent possible observer positions (OPs). Students are introduced to the training environment quickly, and are stepped through using its controls. In Figure 2, we see that the trainee is not clicking on the button that the lesson is asking him to use. If a student persists in such an error, the instructional specification calls for the simulator to make more prominent the object that should be used. In Figure 3, the simulation is highlighting the button in question by flashing a green oval around it.

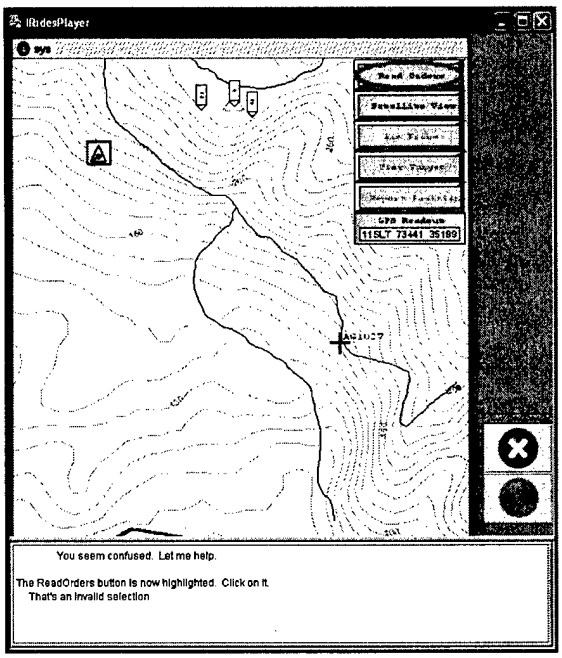

Figure 3. Assisting the Trainee Automatically

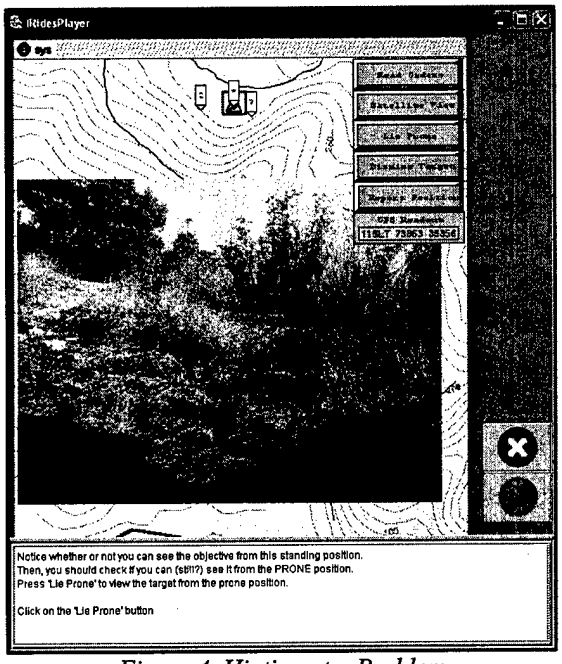

Figure 4. Hinting at a Problem

In Figure 4, the instructional system describes how the forward observer must confirm that the target area is actually in view, especially from the low profile prone position.

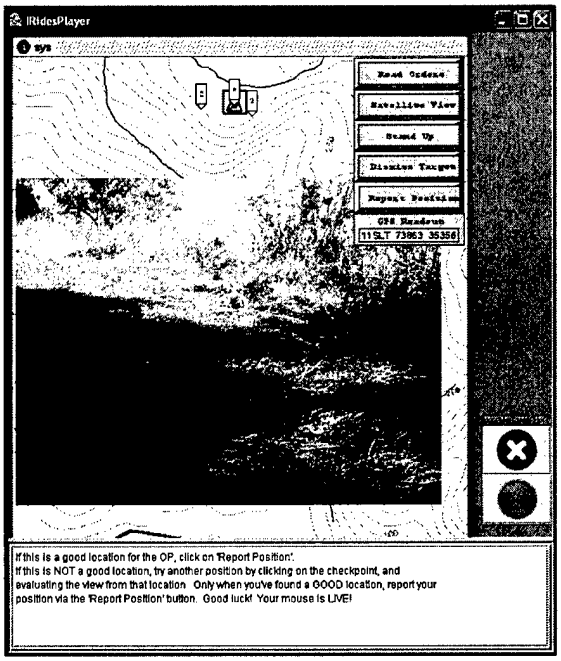

Figure 5. Giving Control to the Trainee

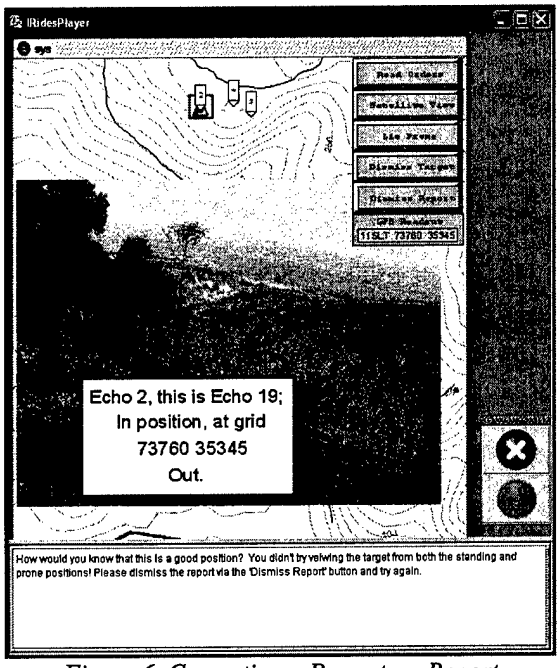

Figure 6. Correcting a Premature Report

After walking the student through the rudiments of the task, the student is given control over the task, as shown in Figure 5. The instructional system then falls into a 'quiet' observational mode, watching the trainee's actions. If the trainee seems to have stopped working, it reminds him or her that, once a usable observation point has been found, the forward observer's position must be reported. When the student actually does report position, the instructional system checks to see whether the position is good. In this prototype trainer, the instructional component wants to know whether the target area is indeed observable, particularly from the prone position, and whether the selected position is exposed to enemy fire. It finds out the answers to these questions by checking with the simulation, which is responsible for tracking these characteristics of possible observation points.

In addition to judging the trainee's actions in light of the simulated context, the instructional system is also able to check that the student has actually performed the requisite actions in the simulation context. In Figure 6 (above), for example, we see that a trainee cannot get away with reporting success without first checking visibility from the prone, more concealed position.

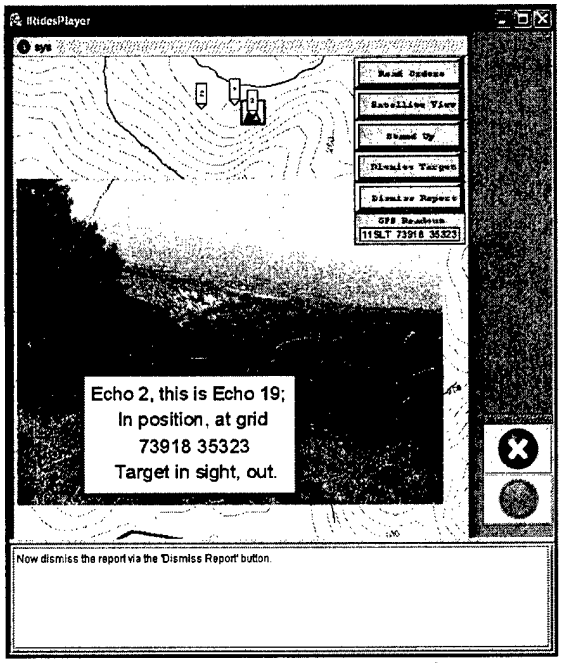

Figure 7. Reporting from a Good OP

In Figure 7, we can see that the student has found a position that gives him a good view of the target area (identified in the target image with the red label). Furthermore, we can see that the trainee has assumed a prone position. (Note that the third button says "Stand Up", rather than "Lie Prone", because the trainee has assumed the prone position.)

If a trainee reports from a bad position (or from a good position, but without sufficient reason), remediation is presented that is specific to that position or to the history of that student's actions in the context of that position. The student is allowed to continue until he or she comes up with a correct observation position and takes the correct actions to ascertain that it is a correct position.

After success on the first problem, the student is given a new forward observer assignment in the same terrain. See Figure 8. The second assignment presents many of the same challenges seen in the first one. But an additional factor can come into play. A trainee can find an observation position that provides a good view of the target area from both standing and prone positions. But it is in a very exposed position, in full view of the target area. Moving just a bit to the northwest puts the observer at a tree line, providing concealment in the shadows. The instructional system catches this problem, should it arise, and points it out to the trainee. See Figure 9.

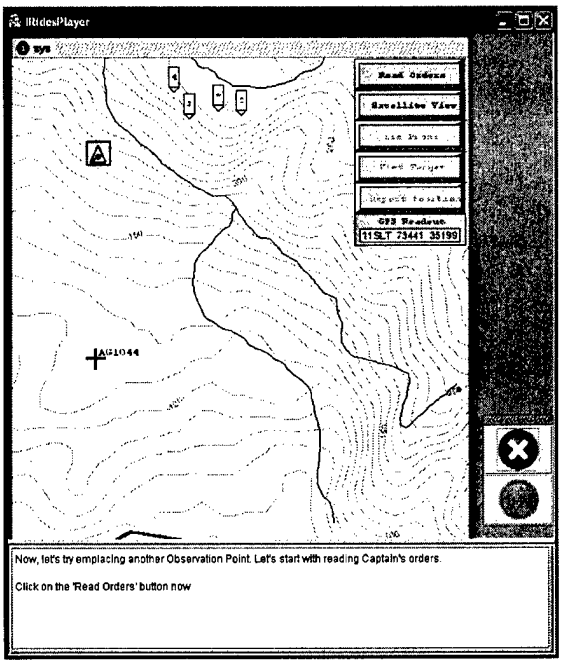

Figure 8. Starting a New Problem

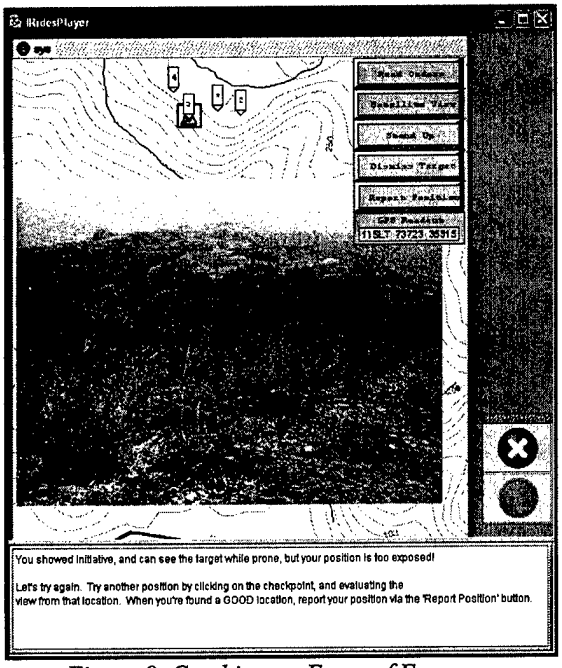

Figure 9. Catching an Error of Exposure

This prototype forward observer exercise demonstrates the feasibility of using an instructional system that can exploit contextual information in a simulation interactively during training. A more nearly complete description of a possible session is shown in Applendix 1.

Related research conducted during the period of this project

During the eight months of this research project, a related effort was conducted under funding from the office of naval research under ONR grant N00014-02-0179 (through sub-award 0070-G-CH640 from the grantee, the Center for the Study of Evaluation at UCLA, to USC's Behavioral Technology Lab). The role of our lab in this grant is to develop advanced tools and techniques for developing and delivering distance learning modules with interactive graphics or simulations. This grant provided support for the following features in iRides, many of which have been exploited in this OIST project.

- Robust authoring and debugging of simulation behavior
  Simulation data editors for objects, attributes, and events were completed. Copy and Paste capabilities were added to these editors. Methods for accessing behavior editors directly from the graphics were introduced. Simple graphics can be promoted to full behaving objects, and objects can be stripped of their rules and demoted to simple graphic status. A debugging interface was developed that supports stepping through the execution of a simulation. Invoked events can be stepped into or completed in one step, under user control. Simulation break points can be inserted. A set of features that support object cloning, including cloning under rule control, was implemented.
- Enhanced Java 2D graphics
  Graphic objects can have transparency, and the degree of transparency can be

controlled interactively by behavior rules. Graphic objects can be given arbitrary images as fill colors. Authors were given the facility to open modal and modeless Java Swing dialogs under simulation rule control, making it possible to develop more professional-looking applications.

- Optional SVG graphics
  The new Scalar Vector Graphics standard (SVG), which is receiving wide support in the application software industry, can now be used to develop certain types of iRides simulations. This makes it possible to use Corel Draw or Adobe Illustrator to develop simulation scenes. Behavior can then be added in the iRides Author application.
- Improved data storage and retrieval
- Advances in iRides instructional capabilities
  The LML markup language and the tutorial controller that interprets it were significantly upgraded to support new capabilities, including making use of several simulations in a single lesson. Improved Question-and-Answer interfaces were designed and implemented. Methods were provided to imitate user actions in the context of a simulation, in order to provide realistic demonstrations. A new approach to authoring instruction using templates was developed. A number of basic templates for instructional interaction were developed, and simple dialog user interfaces were produced for authoring lesson specifications interactively. Classic VIVIDS was also revised so that it could export lessons for iRides using these new templates. The iRides Instructional Document Object Model (DOM) was made w3c compliant, so that lesson specifications can be edited with third-party tools and software modules, An improved method for presenting instructional text in the context of a simulation was developed.

These advanced capabilities can be applied to any advanced Army training requirement that includes low-cost interactive behaviorally complex graphic environments for training, including distance learning.

References

Antal, J.F. (1995) *Infantry Combat: The Rifle Platoon*. Navato, CA: Presidio Press.

Department of the Army, (16 July 1991). *Tactics, Techniques and Procedures for Observed Fire. Field Manual 6-30*, Headquarters, Department of the Army, Washington, D.C., 1991

Johnson, W. L., Rickle, J., Stiles, R. and Munro, A. (1998) Integrating Pedagogical Agents into Virtual Environments. *Presence*.

Munro, A. (1994) Authoring interactive graphical models. In T. de Jong, D. M. Towne, and H. Spada (Eds.), *The Use of Computer Models for Explication, Analysis and Experiential Learning*. Springer Verlag.

Munro, A. (1999) An Open Component Architecture for Simulation-Centered Tutors. In the Proceedings of the Workshop on Advanced Training Technologies and Learning Environments. 9-10 March, NASA Langley Research Center, Hampton, VA.

Munro, A., Johnson, M. C., Pizzini, Q. A., Surmon, D. S., Towne, D. M. and Wogulis, J. L, (1997) Authoring Simulation-Centered Tutors with RIDES. *International Journal of Artificial Intelligence in Education.*, **8**, 284-316.

Munro, A. and Pizzini, Q. A. (1998) *VIVIDS Reference Manual*, Los Angeles: Behavioral Technology Laboratories, University of Southern California.

Munro, A., Pizzini, Q. A., Johnson, M.C., Walker, J., Surmon, D., Dumanoir, P., Garrity, P. (2002). On-demand interactive simulation-based training (OIST), Abstract in the Proceedings of the Army Science Conference.

Munro, A., Surmon, D., Johnson, M., Pizzini, Q., and Walker, J. (1999) An Open Architecture for Simulation-Centered Tutors. In Lajoie, S. P. and Vivet, M., *Artificial Intelligence in Education: Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration*, Amsterdam: IOS Press, 360-367.

Pizzini, Q.A., Munro, A., Wogulis, J.L., and Towne, D.M., (1996). The Cost-Effective Authoring of Procedural Training, in *Architectures and Methods for Designing Cost-Effective and Reusable ITSs* Workshop Proceedings, ITS'96, Montreal, Québec, Canada.

Poole, H. J. (1998) Evening Up the Odds with Indirect Fire, *The Last Hundred Yards: The NCO's Contribution to the Art of War,* Posterity Press

Appendix 1: A Walkthrough of a Forward Observer Training Session

The screenshots in this appendix portray certain steps during a session with the iRides Forward Observer trainer. There are three major sections in this training session. In the first, the training system explains the task and guides the trainee in the use of the system. In the second section, the trainee completes the first problem and receives instructional feedback and guidance in response. In the last section, a new problem is posed and the student works on it and receives feedback. When a trainee does not handle a problem correctly, he or she is given the opportunity to continue, in order to search for a better solution.

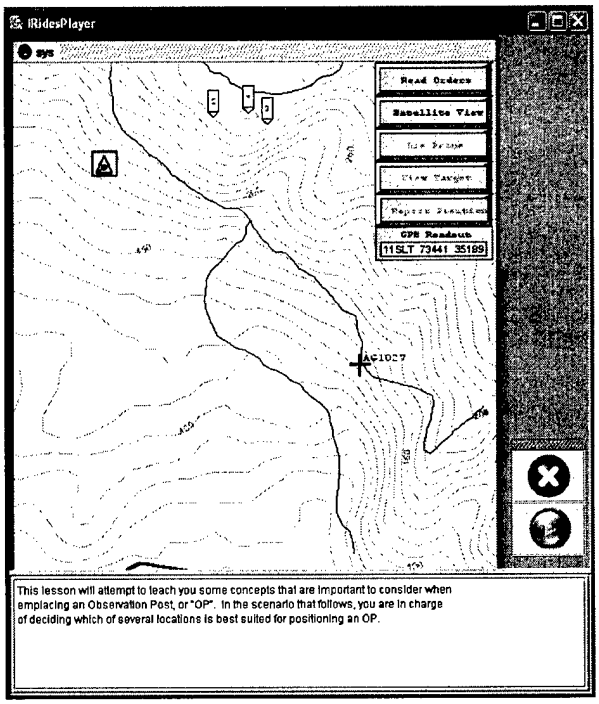

Figure 1.

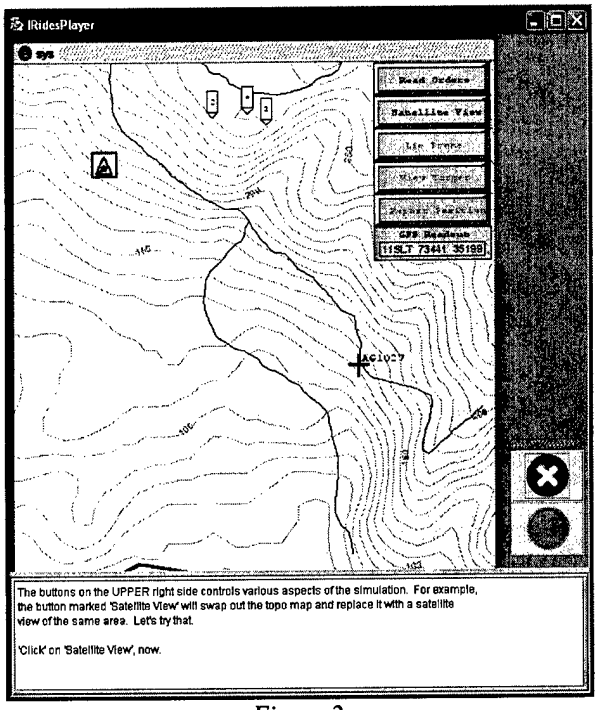

Figure 2.

In Figure 1, the first view that a trainee sees is displayed. The trainee is informed that the task is to select an observation post among several candidates. After sufficient time to read these sentences has passed, text appears that explains the function of the "Continue" button—the silver button below the red "X" quit button at the lower right of these screenshots. Next, as shown in Figure 2, the lesson explains the simulation interface controls, the buttons at the upper right in these figures. Here the trainee is instructed to click the "Satellite View" button. Once he or she does so, the topographic map view is replaced with a composite satellite photo view, as shown in Figure 3, below. The trainee is then guided to return to the topographic view, as shown in Figure 4. The training system then explains the symbology used in this lesson. The triangle in a square represents the position of the trainee, the forward observer. The symbols that look like vertical rectangles above downward pointing triangles are the numbered candidate observation posts. The assigned location is shown with a "*" symbol, rather than with the number "1" in its box.

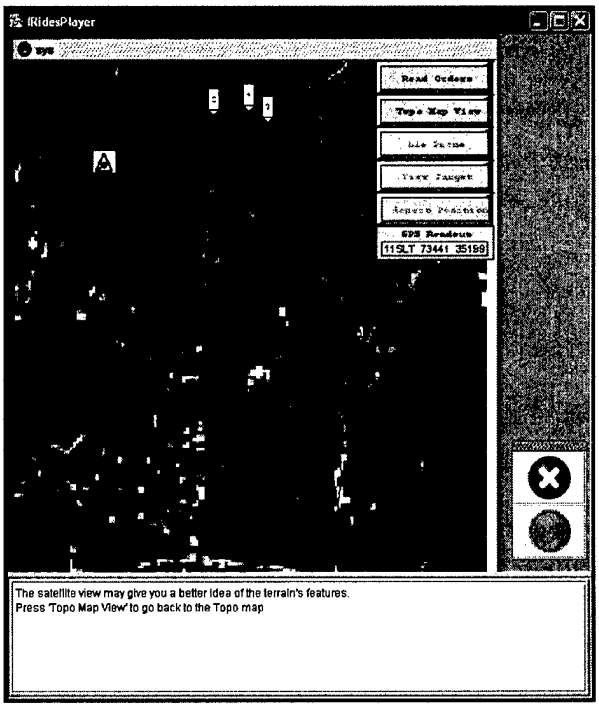

Figure 3.

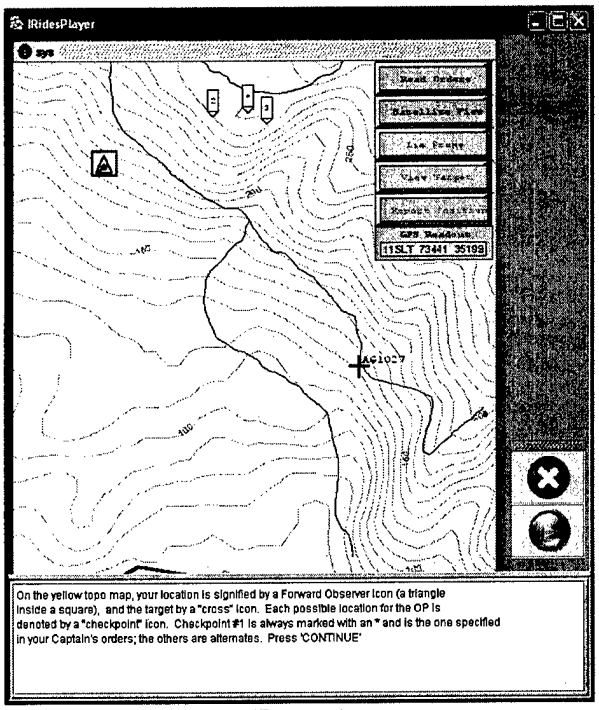

Figure 4.

Automatic Assistance. Trainees are taught (Figure 5) to use the "Read Orders" button.

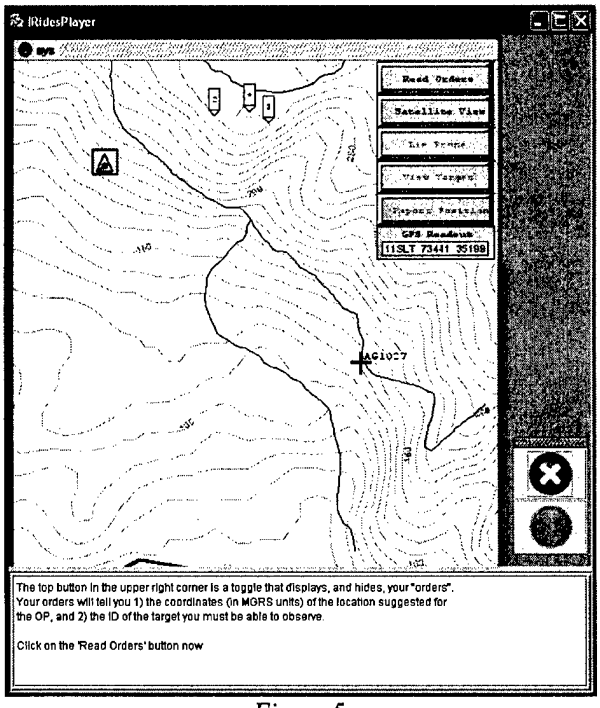

Figure 5.

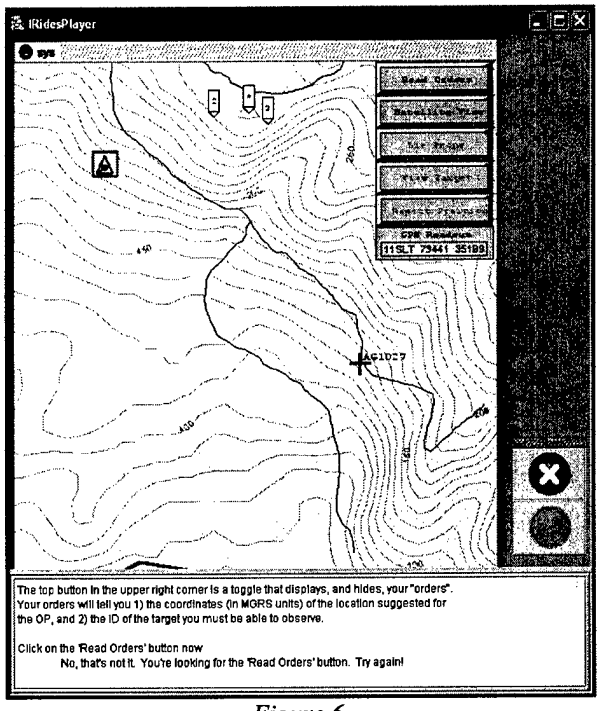

Figure 6.

During this part of the lesson, when the instructional system asks the student to carry out an action it requests that the simulator not carry out user actions directly, but that it instead inform the instruction about what was selected. If the correct object is selected,

the simulator is allowed to respond as it would normally. If an incorrect object is selected, the simulation behavior component is not informed, so the user action has no effect in the simulation. The instructional script, however, takes note, and gives the student corrective feedback, repeating the request, as in Figure 6, where the trainee is asked again to click on the "Read Orders" button. If a student persists in not carrying out a directive correctly, the trainer will automatically highlight the object that is the target. In Figure 7, the "Read Orders" button is circled with a flashing green ellipse. When the student does finally carry out the action correctly, the action is allowed to take effect in the simulated world. In Figure 8, we see that the orders for this forward observer problem have been brought up (as a result of the student finally pressing the correct button).

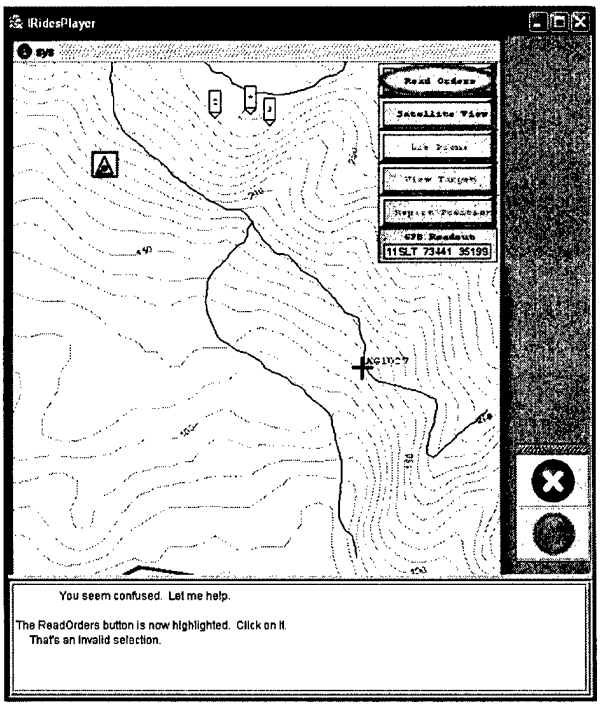

Figure 7.

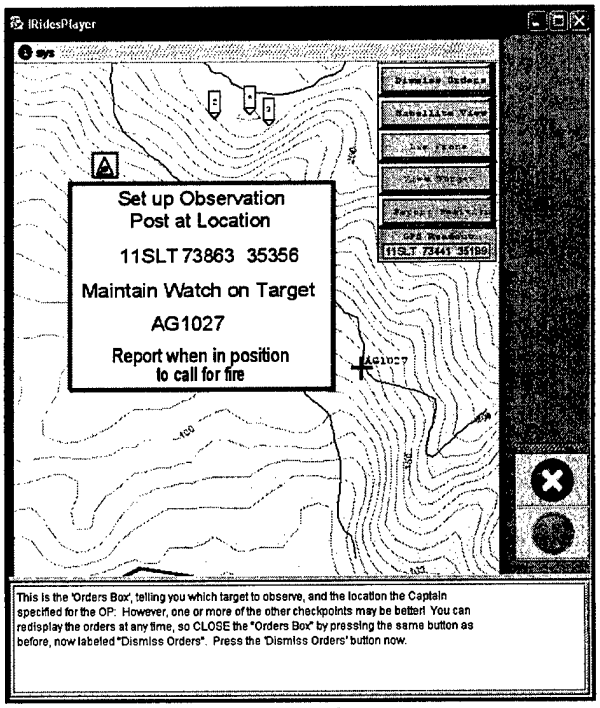

Figure 8.

Trainees read that they have been ordered to set up an observation post at a particular location in order to maintain watch on a specific target area. The observation post is given in the Military Grid Reference System (MGRS). As the observer icon moves from one position to another on the screen, the GPS Readout, shown below the buttons, changes to provide an updated MGRS position.

The trainee is then instructed to choose the ordered Observation Post (OP), by clicking on the position with the "*" in its symbol. In Figure 10, the instructional system points out that, when the observer icon arrives at the assigned OP position, some of the control buttons become enabled. It directs the user to choose the "View Target" button in order to find out what can be seen from that position. See Figure 11 for the view from the ordered position. The trainer points out that the view should be checked out from the prone position as well. Figure 12 shows that view from the position given in the orders. Because this position is too far back of the military crest, it is not possible to see the target area assigned. A forward observer must be aware of the intent of his orders.

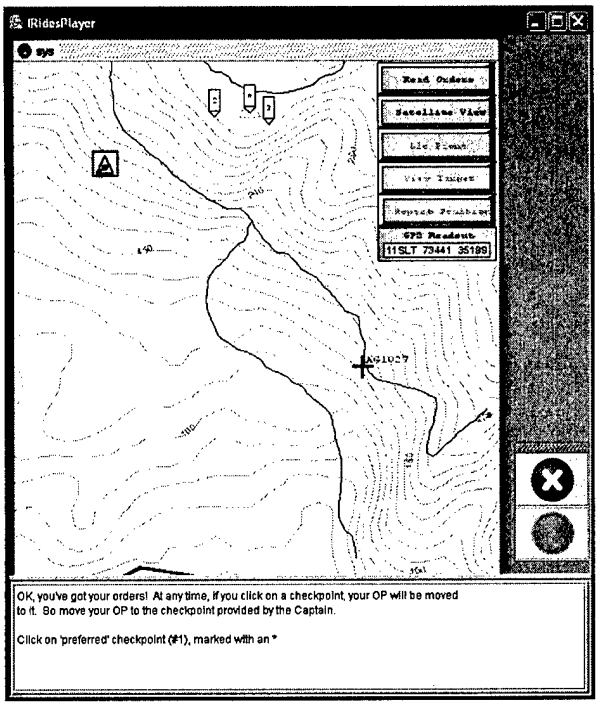

Figure 9.

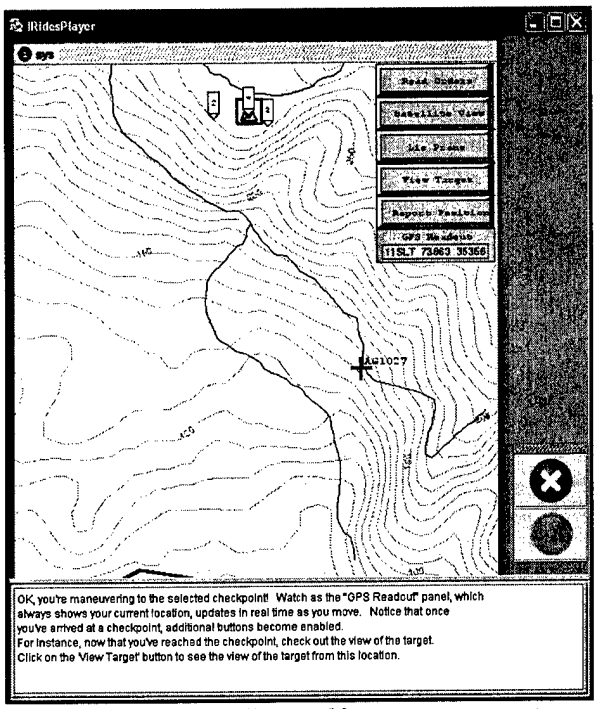

Figure 10.

Since the area to be observed is not visible, a different position for the OP must be found.

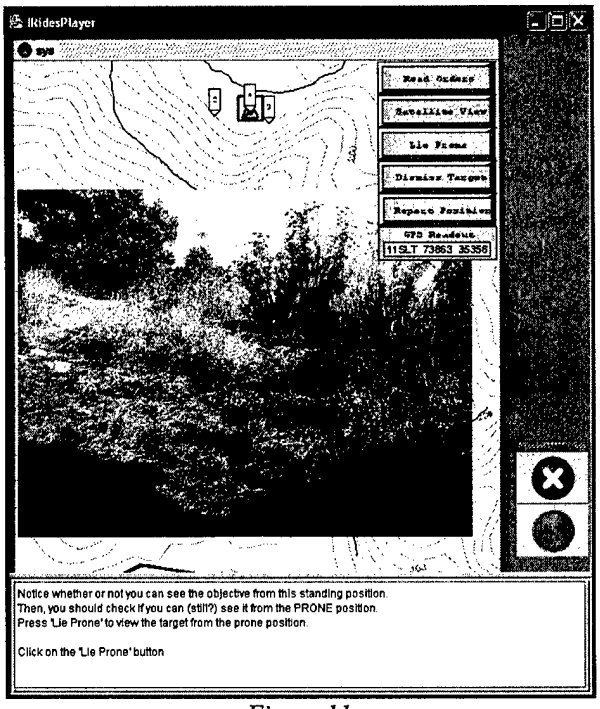

Figure 11.

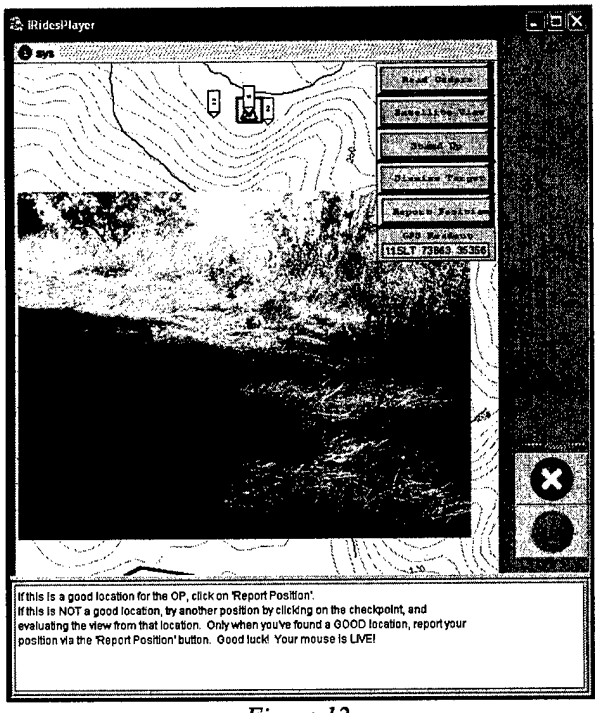

Figure 12.

At this point, the trainee is told that the task of finding a good OP is now in his hands. When a good position is found, it must be reported using the "Report Position" button.

Control is now in the hands of the student. From time to time, the system reminds the student that the goal is to report from a good position. See Figure 13.

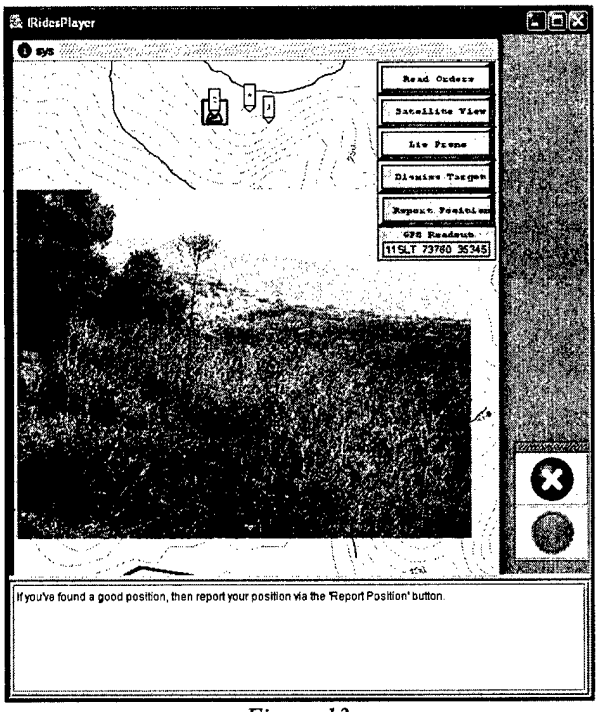

Figure 13.

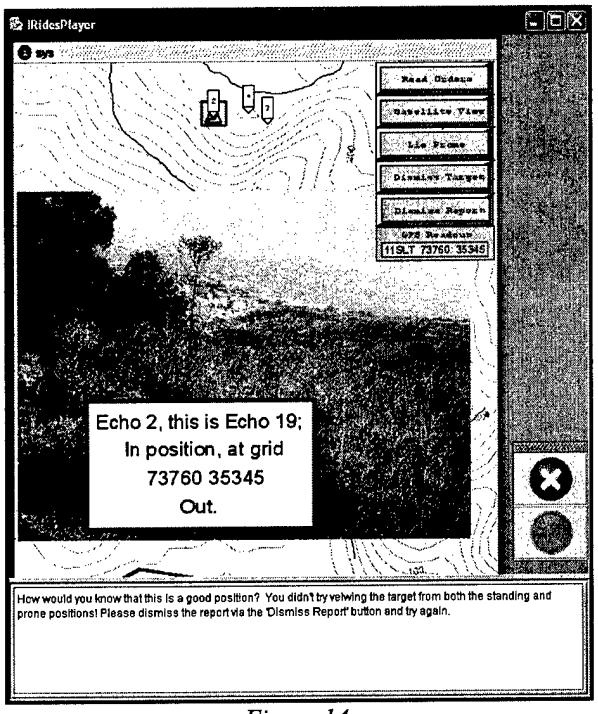

Figure 14.

In Figure 14, the trainee has decided to report from this candidate position #2, which has an excellent view of the target area AG1027. The instruction system has been watching

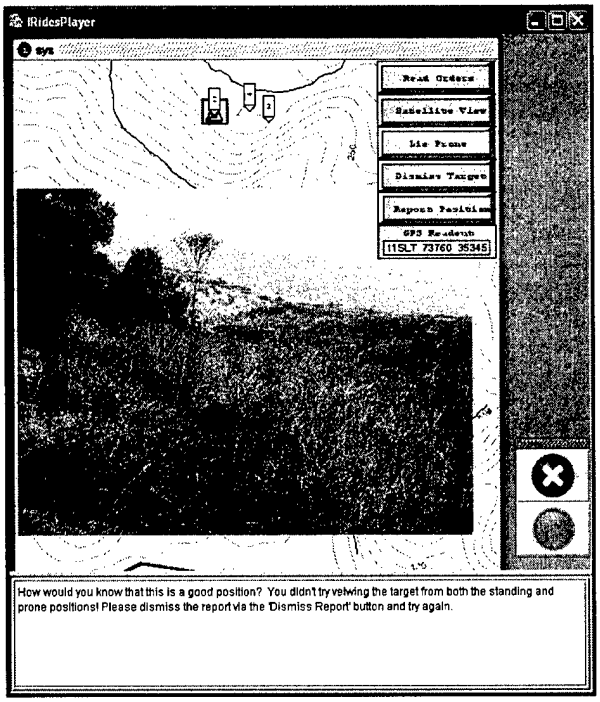

Figure 15.

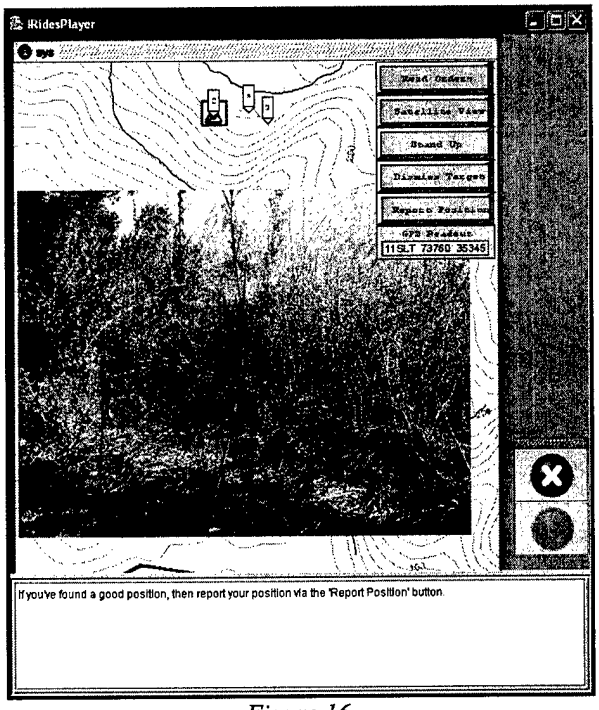

Figure 16.

the student and knows that the position has not been evaluated from the prone position. See Figure 15. The prone view is shown in Figure 16.

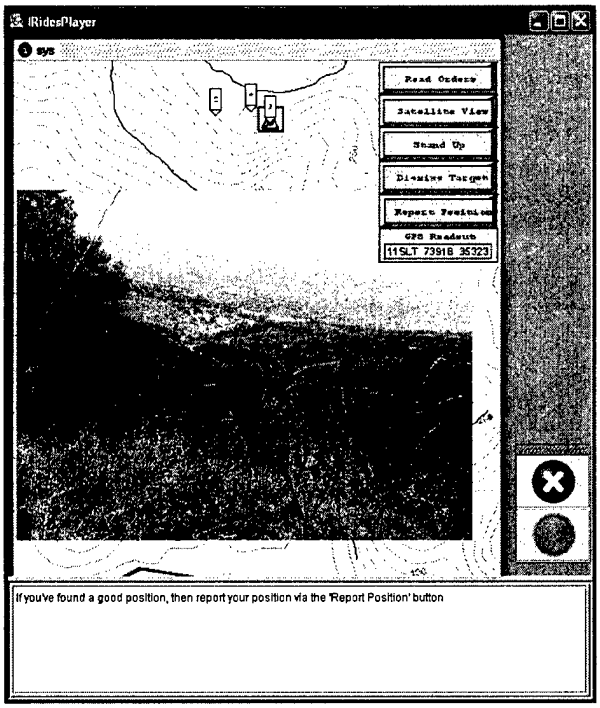

Figure 17.

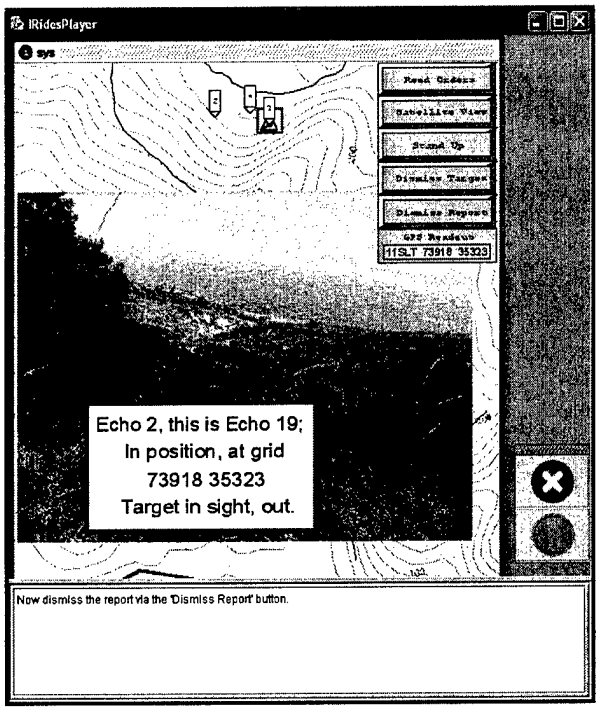

Figure 18.

In Figures 17, the trainee checks the view of the target area from the third reference point, in both the standing and prone positions. The target area can be viewed. At this point the student reports the position, as shown in Figure 18, and is prompted to clear the report display. Next this 'solution' is evaluated. The trainer says "Excellent. You showed initiative and found the ideal spot, and can see the target while prone." See Figure 19.

Trainees must realize that micro-terrain features, such as the placement of small boulders, brush, trees, and land features such as the military crest and other topographic features may require adjusting the assigned position to one that can be used to carry out the observation assignment. More important than carrying out a literal order is to understand the purpose of that order and to carry out the order in such a way that its purpose is achieved.

The trainer then prompts the student to press the Continue button, and a new forward observation case is presented. See Figure 20.

The trainee is then told to use the View Orders button to find out what is required for this problem. The orders (Figure 21) direct the observation of a downslope area to the West. In Figure 22, the trainee has moved to the assigned position and must decide whether to report.

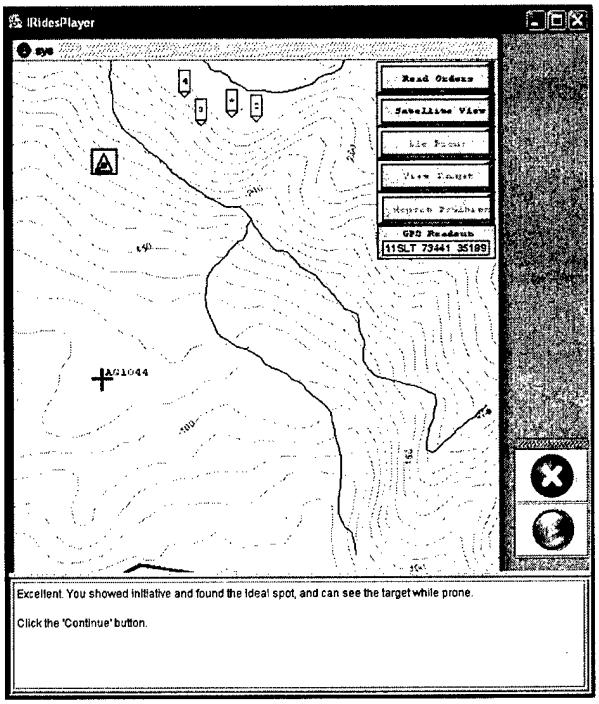

Figure 19.

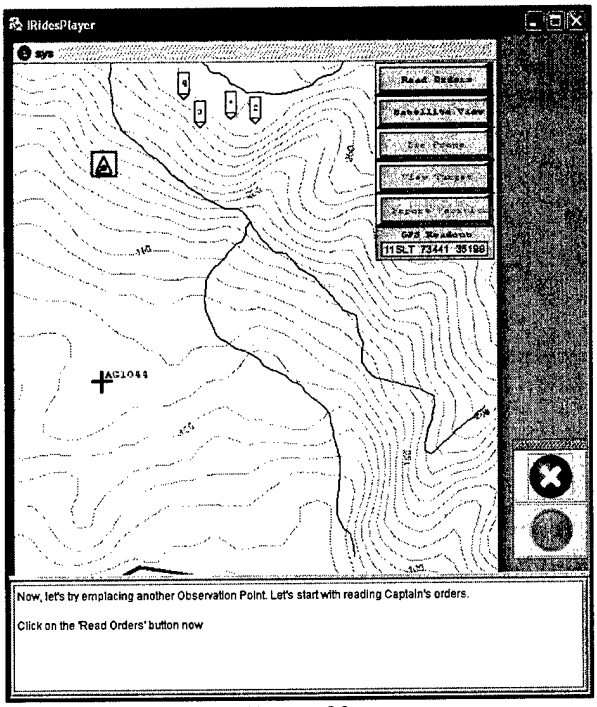

Figure 20.

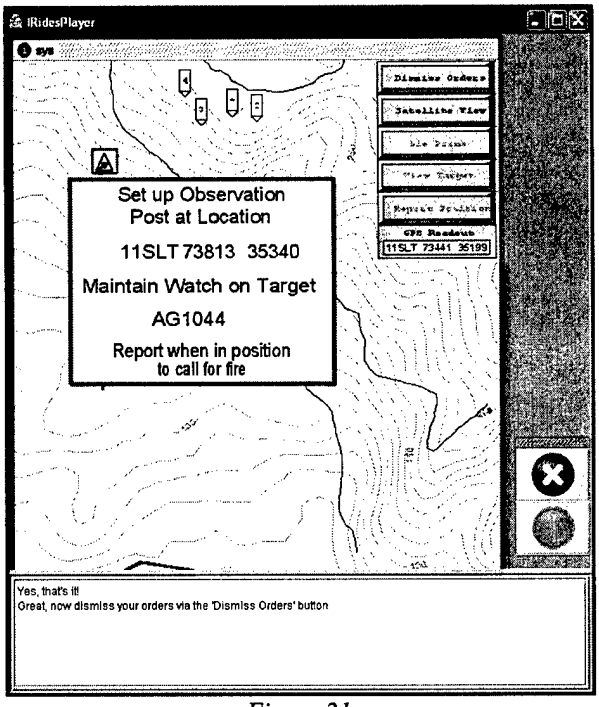

Figure 21.

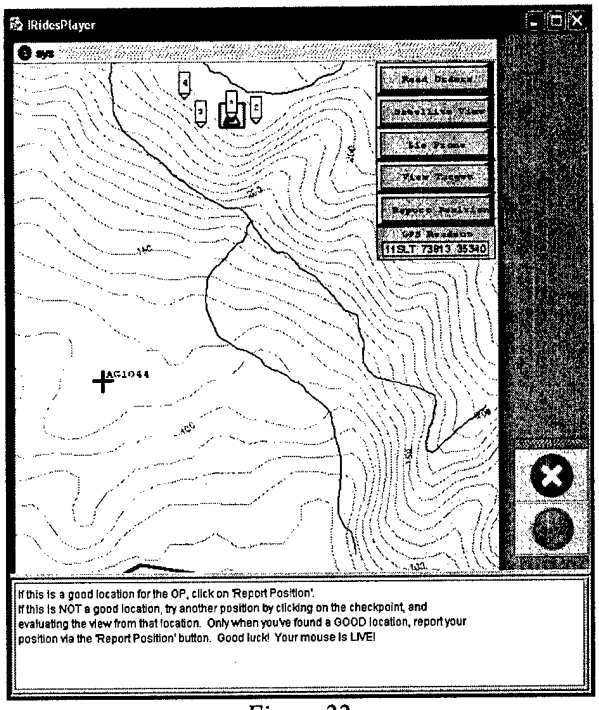

Figure 22.

Again, as can be seen in Figure 23, there are terrain features that prevent viewing the assigned target. After finding problems with the prone position at the second checkpoint, the trainee moves to the third position. Here the target area is visible from both the standing position and when in the prone position. See Figure 24. So the student reports the position.

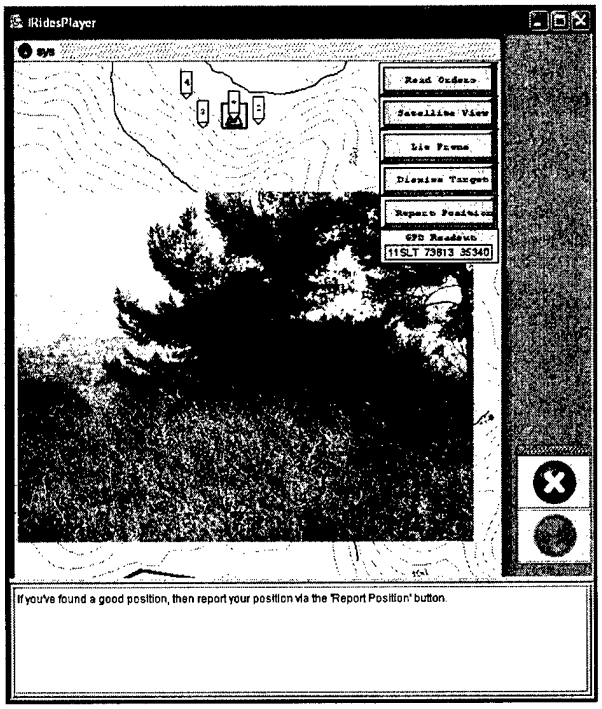

Figure 23.

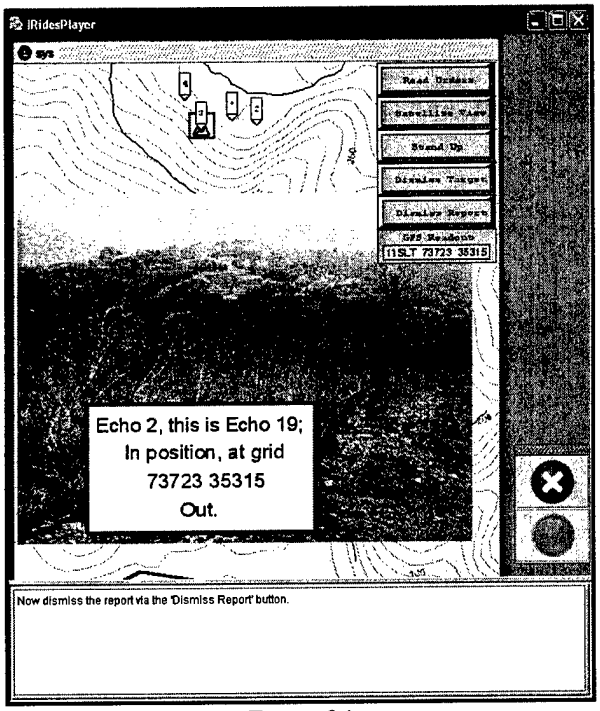

Figure 24.

Unfortunately, this position has a different kind of problem. The observer is out in the open, exposed to view from the target area. See the remediation in Figure 25.

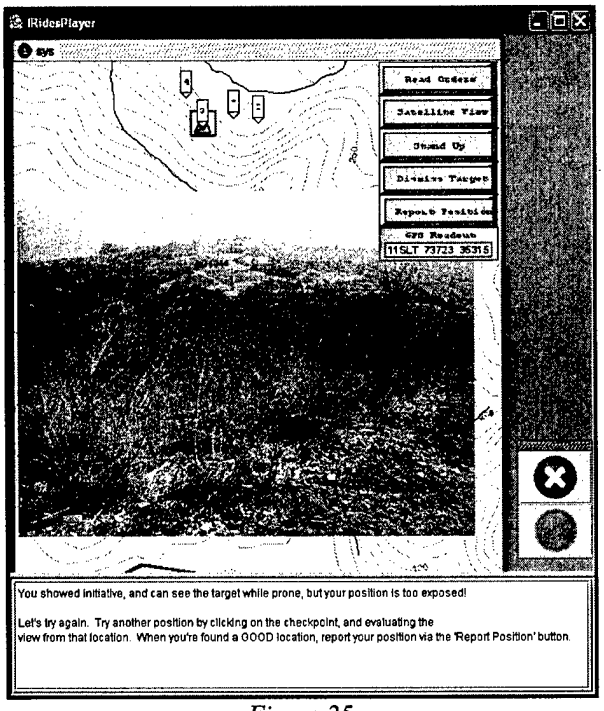

Figure 25.

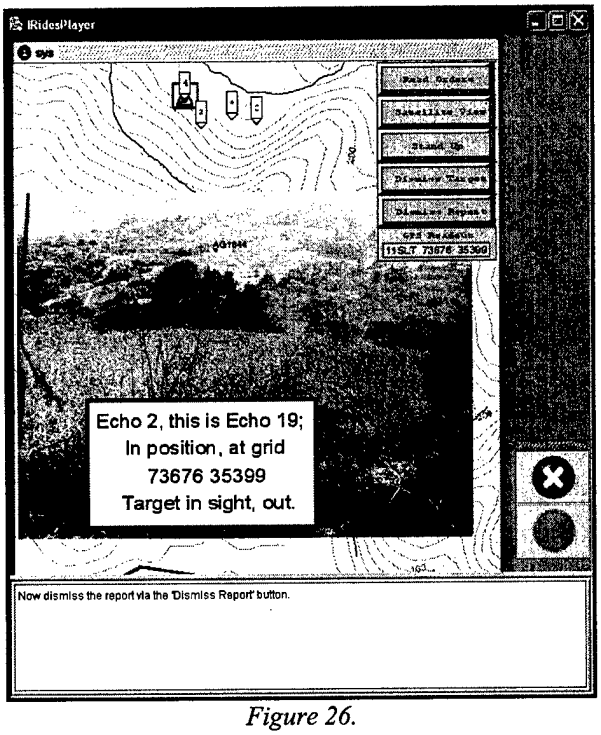

Figure 26.

In Figure 26, the trainee is viewing the target area from the fourth checkpoint. Not only is the target area in view, but the position is upslope from the previous position, at the edge

of a tree line. The shadows on the ground in this image are cast by tree branches hanging over the observer. The combination of being in the shadow and having some concealment from plants make this a much less exposed position from which to observe.

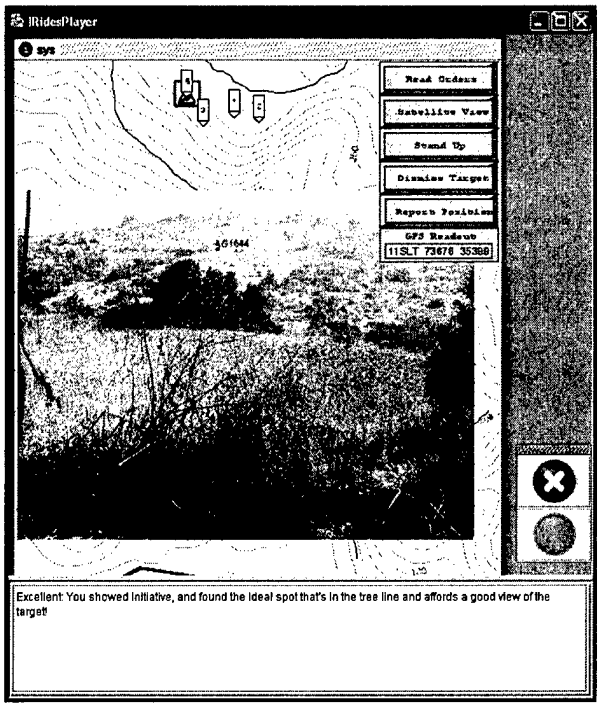

Figure 27.

Finally, the trainee is complemented for finding a position that offers both a good view of the target area and some concealment.

Appendix 2: The LML Script in Control of the Instruction

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE lml SYSTEM "lessonML.dtd">

<lml name="lesson1"
onActive="score=0.0;possibleScore=0.0;thisScore=0.0">
  <group>
            <script href="mjUserMouse.js"/>
    <use href="Tstartup.lml#VStartup">
      <setProperty name="Simulation" value="microT.jr"/>
    </use>

            <!-- Intro -->
    <present presentation="text">
This lesson will attempt to teach you some concepts that are important
to consider when
emplacing an Observation Post, or "OP".  In the scenario that follows,
you are in charge
of deciding which of several locations is best suited for positioning
an OP.
    </present>
    <requireCommand exitCommands="continue" timeAllotted="30" />

            <!-- What is the 'Continue' button -->
    <present presentation="text"
onReady="umb.clearPresentation('text')">
But first, you need to know what the buttons in the upper and lower
right corners are for.
The buttons on the LOWER right side control the lesson.  The red button
is the 'Quit' button,
and below that is the gray 'Continue' button.    Press the CONTINUE
button now.
    </present>
    <requireCommand exitCommands="continue"/>

    <!-- Satellite View On -->
    <present presentation="text"
onReady="umb.clearPresentation('text')">
The buttons on the UPPER right side controls various aspects of the
simulation.  For example,
the button marked 'Satellite View' will swap out the topo map and
replace it with a satellite
view of the same area.  Let's try that.
    </present>
    <use href="TfindMJ.lml#VFind">
      <setProperty name="Object"  value=".sys.ctrlPanel.ButtonTopo."/>
      <setProperty name="ObjectName"  value="Satellite View button"/>
      <setProperty name="Text"  value="'Click' on 'Satellite View',
now."/>
      <setProperty name="IncorrectStillTryingText"  value="No, that's
not it.  You're looking for the 'Satellite View' button.  Try again!"/>
      <setProperty name="CorrectText"  value="Yes, that'll bring up the
Satellite view." />
    </use>
            <code>
```

```
                quickclick(".sys.ctrlPanel.ButtonTopo.");
            </code>

             <!-- Satellite View Off -->
    <present presentation="text"
onReady="umb.clearPresentation('text')">
The satellite view may give you a better idea of the terrain's
features.
Press 'Topo Map View' to go back to the Topo map.
    </present>
    <use href="TfindMJ.lml#VFind">
      <setProperty name="Object"  value=".sys.ctrlPanel.ButtonTopo."/>
      <setProperty name="ObjectName"  value="Satellite View button"/>
      <setProperty name="Text"  value=" "/>
      <setProperty name="IncorrectStillTryingText"  value="No, that's
not it.  You're looking for the 'Topo Map View' button.  Try again!"/>
      <setProperty name="CorrectText"  value="Yes, that'll restore the
Topographic Map." />
    </use>
            <code>
                quickclick(".sys.ctrlPanel.ButtonTopo.");
            </code>

  <present presentation="text" onReady="umb.clearPresentation('text')">
On the yellow topo map, your location is signified by a Forward
Observer icon (a triangle
inside a square),  and the target by a "cross" icon.  Each possible
location for the OP is
denoted by a "checkpoint" icon.  Checkpoint #1 is always marked with an
* and is the one specified
in your Captain's orders; the others are alternates.  Press 'CONTINUE'
     </present>
     <requireCommand exitCommands="continue"/>

    <!-- Read Orders demo -->
     <present presentation="text"
onReady="umb.clearPresentation('text')">
The top button in the upper right corner is a toggle that displays, and
hides, your "orders".
Your orders will tell you 1) the coordinates (in MGRS units) of the
location suggested for
the OP, and 2) the ID of the target you must be able to observe.
</present>

    <use href="TfindMJ.lml#VFind">
      <setProperty name="Object"
value=".sys.ctrlPanel.ButtonOrders."/>
      <setProperty name="ObjectName"  value="ReadOrders button"/>
      <setProperty name="CorrectText"  value="Yes, that's it!" />
      <setProperty name="IncorrectStillTryingText"  value="No, that's
not it.  You're looking for the 'Read Orders' button.  Try again!"/>
      <setProperty name="Text"  value="Click on the 'Read Orders'
button now"/>
    </use>
            <code>
                quickclick(".sys.ctrlPanel.ButtonOrders.");
            </code>
```

```
    <present presentation="text"
onReady="umb.clearPresentation('text')">
This is the 'Orders Box', telling you which target to observe, and the
location the Captain
specified for the OP:  However, one or more of the other checkpoints
may be better!  You can
redisplay the orders at any time, so CLOSE the "Orders Box" by pressing
the same button as
before, now labeled "Dismiss Orders".  Press the 'Dismiss Orders'
button now.</present>

    <!-- Dismiss Orders demo -->
    <use href="TfindMJ.lml#VFind">
      <setProperty name="Object"
value=".sys.ctrlPanel.ButtonOrders."/>
      <setProperty name="ObjectName"  value="ReadOrders button"/>
      <setProperty name="Text"  value=" "/>
      <setProperty name="CorrectText"  value="Yes, that's it!" />
      <setProperty name="IncorrectStillTryingText"  value="No, that's
not it.  You're looking for the 'Dismiss Orders' button.  Try again!"/>
    </use>
            <code>
                  quickclick(".sys.ctrlPanel.ButtonOrders.");
            </code>

    <present presentation="text"
onReady="umb.clearPresentation('text')">
OK, you've got your orders!  At any time, if you click on a checkpoint,
your OP will be moved
to it.  So move your OP to the checkpoint provided by the Captain.
    </present>

            <!-- Move To Checkpoint * demo -->
    <use href="TfindMJ.lml#VFind">
      <setProperty name="Object"  value=".sys.point1."/>
      <setProperty name="ObjectName"  value="Suggested checkpoint
(1)"/>

      <setProperty name="Text"  value="Click on 'preferred' checkpoint
(#1), marked with an *"/>
      <setProperty name="CorrectText"  value="Yes, that's it!" />
      <setProperty name="IncorrectStillTryingText"  value="No, you need
to click the checkpoint marked with an asterisk *"/>
    </use>
            <code>
                  quickclick(".sys.point1.");
            </code>

    <present presentation="text"
onReady="umb.clearPresentation('text')">
OK, you're maneuvering to the selected checkpoint!  Watch as the "GPS
Readout" panel, which
always shows your current location, updates in real time as you move.
Notice that once
you've arrived at a checkpoint, additional buttons become enabled.
</present>
```

```
    <!-- wait till at a checkpoint -->
            <requireGoal expression=".sys.iconOP.atPoint != -1"/>

            <!-- View Target demo -->
            <present presentation="text">
For instance, now that you've reached the checkpoint, check out the
view of the target. </present>
    <use href="TfindMJ.lml#VFind">
      <setProperty name="Object"
value=".sys.ctrlPanel.ButtonLookTarget."/>
      <setProperty name="ObjectName"  value="View Target Button"/>
      <setProperty name="Text"  value="Click on the 'View Target'
button to see the view of the target from this location."/>
      <setProperty name="CorrectText"  value="Yes, that's it!" />
      <setProperty name="IncorrectStillTryingText"  value="No, you need
to click the 'View Target' button!"/>
    </use>
            <code>
                  quickclick(".sys.ctrlPanel.ButtonLookTarget.");
            </code>

            <!-- Lay Prone demo -->
    <present presentation="text"
onReady="umb.clearPresentation('text')">
Notice whether or not you can see the objective from this standing
position.
Then, you should check if you can (still?) see it from the PRONE
position.
Press 'Lie Prone' to view the target from the prone position.
    </present>
    <use href="TfindMJ.lml#VFind">
      <setProperty name="Object"  value=".sys.ctrlPanel.ButtonStand."/>
      <setProperty name="ObjectName"  value="lay prone button"/>
      <setProperty name="Text"  value="Click on the 'Lie Prone'
button"/>
      <setProperty name="CorrectText"  value="Yes, that's it!" />
      <setProperty name="IncorrectStillTryingText"  value="No, that's
not it.  Press the 'Lie Prone' button."/>
    </use>
            <code>
                  quickclick(".sys.ctrlPanel.ButtonStand.");
            </code>

            <!-- Wrap up, and cut 'em loose! -->
    <present presentation="text"
onReady="umb.clearPresentation('text')">
If this is a good location for the OP, click on 'Report Position'.
If this is NOT a good location, try another position by clicking on the
checkpoint, and
evaluating the view from that location.  Only when you've found a GOOD
location, report your
position via the 'Report Position' button.  Good luck!  Your mouse is
LIVE!
    </present>
    <do condition="umb.evaluateExpression('.sys.evaledCorrect != true
|| .sys.triedProne != true')">
```

```
      <do
condition="umb.evaluateExpression('.sys.ctrlPanel.ButtonReport.reported
!= true')">
        <requireGoal name="reportbutton" onReady="score=NaN"
expression=".sys.ctrlPanel.ButtonReport.reported == true"
timeAllotted="30" />
        <conditional
condition="umb.evaluateExpression('.sys.ctrlPanel.ButtonReport.reported
!= true')">
          <present presentation="text"
onReady="umb.clearPresentation('text')">
If you've found a good position, then report your position via the
'Report Position' button.
          </present>
          <group>
            <conditional
condition="umb.evaluateExpression('.sys.triedProne != true')">
              <group>
                <present presentation="text"
onReady="umb.clearPresentation('text')">
How would you know that this is a good position?  You didn't try
veiwing the target from both the standing and  prone positions! Please
dismiss the report via the 'Dismiss Report' button and try again.
                </present>
                <do condition="true">
                  <requireGoal name="dismissbutton2"
onReady="score=NaN" expression=".sys.needToEvaluateChoice == true"
timeAllotted="30"/>
                <conditional condition="isNaN(previousSibling.score)">
                    <present presentation="text"
onReady="umb.clearPresentation('text')">
Please dismiss the report via the 'Dismiss Report' button and then
report your position after viewing target from both the standing and
biewing positions.
                    </present>
                    <break/>
                  </conditional>
                </do>
                <break/>
              </group>
              <group>
                <present presentation="text"
onReady="umb.clearPresentation('text')">
Now dismiss the report via the 'Dismiss Report' button.
                </present>
                <do onInitialized="this.done = false" condition="true">
                  <requireGoal name="dismissbutton" onReady="score=NaN"
expression=".sys.needToEvaluateChoice == true" timeAllotted="30"/>
                <conditional condition="isNaN(previousSibling.score)">
                    <present presentation="text"
onReady="umb.clearPresentation('text')">
Please dismiss the report via the 'Dismiss Report' button.
                    </present>
                    <group>
                      <present presentation="text"
onReady="umb.clearPresentation('text')">
<eval>umb.getAttributeValue(".sys.evaledRemediation")</eval>
```

```
                        </present>
                        <conditional
condition="umb.evaluateExpression('.sys.evaledCorrect != true')">
                          <present presentation="text">
Let's try again.  Try another position by clicking on the checkpoint,
and evaluating the
view from that location.  When you're found a GOOD location, report
your position via the 'Report Position' button.
                          </present>
                        </conditional>
                        <break/>
                      </group>
                    </conditional>
                  </do>
                </group>
              </conditional>
              <break/>
            </group>
          </conditional>
        </do>
    </do>
    <code>
       umb.setAttributeValue(".sys.endOfProblem",true);
    </code>
    <present presentation="text">
Click the 'Continue' button.
    </present>
    <requireCommand exitCommands="continue"/>
    <present presentation="text"
onReady="umb.clearPresentation('text')">
Now, let's try emplacing another Observation Point. Let's start with
reading Captain's orders.
  </present>
     <use href="TfindMJ.lml#VFind">
       <setProperty name="Object"
value=".sys.ctrlPanel.ButtonOrders."/>
       <setProperty name="ObjectName"  value="ReadOrders button"/>
       <setProperty name="CorrectText"  value="Yes, that's it!" />
       <setProperty name="IncorrectStillTryingText"  value="No, that's
not it.  You're looking for the 'Read Orders' button.  Try again!"/>
       <setProperty name="Text"  value="Click on the 'Read Orders'
button now"/>
     </use>
             <code>
                   quickclick(".sys.ctrlPanel.ButtonOrders.");
             </code>
     <present presentation="text">
Great, now dismiss your orders via the 'Dismiss Orders' button.
     </present>

     <!-- Dismiss Orders demo -->
     <use href="TfindMJ.lml#VFind">
       <setProperty name="Object"
value=".sys.ctrlPanel.ButtonOrders."/>
       <setProperty name="ObjectName"  value="ReadOrders button"/>
       <setProperty name="Text"  value=" "/>
       <setProperty name="CorrectText"  value="Yes, that's it!" />
```

```
      <setProperty name="IncorrectStillTryingText"  value="No, that's
not it.  You're looking for the 'Dismiss Orders' button.  Try again!"/>
    </use>
            <code>
                  quickclick(".sys.ctrlPanel.ButtonOrders.");
            </code>

            <!-- Move To Checkpoint * demo -->
    <use href="TfindMJ.lml#VFind">
      <setProperty name="Object"  value=".sys.point1."/>
      <setProperty name="ObjectName"  value="Suggested checkpoint
(1)"/>
      <setProperty name="Text"  value="Click on 'preferred' checkpoint
(#1), marked with an *"/>
      <setProperty name="CorrectText"  value="Yes, that's it!" />
      <setProperty name="IncorrectStillTryingText"  value="No, you need
to click the checkpoint marked with an asterisk *"/>
    </use>
    <code>
      quickclick(".sys.point1.");
    </code>
    <present presentation="text"
onReady="umb.clearPresentation('text')">
If this is a good location for the OP, click on 'Report Position'.
If this is NOT a good location, try another position by clicking on the
checkpoint, and
evaluating the view from that location.  Only when you've found a GOOD
location, report your
position via the 'Report Position' button.  Good luck!  Your mouse is
LIVE!
    </present>
    <do  condition="umb.evaluateExpression('.sys.evaledCorrect != true
|| .sys.triedProne != true')">
      <do
condition="umb.evaluateExpression('.sys.ctrlPanel.ButtonReport.reported
!= true')">
        <requireGoal name="reportbutton2" onReady="score=NaN"
expression=".sys.ctrlPanel.ButtonReport.reported == true"
timeAllotted="30" />
        <conditional
condition="umb.evaluateExpression('.sys.ctrlPanel.ButtonReport.reported
!= true')">
          <present presentation="text"
onReady="umb.clearPresentation('text')">
If you've found a good position, then report your position via the
'Report Position' button.
          </present>
          <group>
            <conditional
condition="umb.evaluateExpression('.sys.triedProne != true')">
              <group>
                <present presentation="text"
onReady="umb.clearPresentation('text')">
How would you know that this is a good position?  You didn't try
veiwing the target from both the standing and  prone positions! Please
dismiss the report via the 'Dismiss Report' button and try again.
                </present>
```

```
                <do name="do4b" condition="true">
                  <requireGoal name="dismissbutton2b"
onReady="score=NaN" expression=".sys.needToEvaluateChoice == true"
timeAllotted="30"/>
                <conditional condition="isNaN(previousSibling.score)">
                    <present presentation="text"
onReady="umb.clearPresentation('text')">
Please dismiss the report via the 'Dismiss Report' button and then
report your position after viewing target from both the standing and
biewing positions.
                    </present>
                    <break/>
                  </conditional>
                </do>
                <break/>
              </group>
              <group>
                <present presentation="text"
onReady="umb.clearPresentation('text')">
Now dismiss the report via the 'Dismiss Report' button.
                </present>
                <do name="do3b" onInitialized="this.done = false"
condition="true">
                  <requireGoal name="dismissbutton" onReady="score=NaN"
expression=".sys.needToEvaluateChoice == true" timeAllotted="30"/>
                <conditional condition="isNaN(previousSibling.score)">
                    <present presentation="text"
onReady="umb.clearPresentation('text')">
Please dismiss the report via the 'Dismiss Report' button.
                    </present>
                    <group>
                      <present presentation="text"
onReady="umb.clearPresentation('text')">
<eval>umb.getAttributeValue(".sys.evaledRemediation")</eval>
                      </present>
                      <conditional
condition="umb.evaluateExpression('.sys.evaledCorrect != true')">
                        <present presentation="text">
Let's try again.  Try another position by clicking on the checkpoint,
and evaluating the
view from that location.  When you're found a GOOD location, report
your position via the 'Report Position' button.
                        </present>
                      </conditional>
                      <break/>
                    </group>
                  </conditional>
                </do>
              </group>
            </conditional>
            <break/>
          </group>
        </conditional>
      </do>
    </do>
  </group>
</lml>
```

Appendix 3: The *Call For Fire* Task

Indirect fire is a great equalizer: A small squad can often neutralize a large force by expedient and accurate use of artillery. But to do so, the forward observer must be well trained, and recently trained, because the call to fire is a perishable skill.

According to recent Army doctrine[1], there are six elements in the message sent when calling for fire over a voice network:

1. Observer Identification
2. Warning Order
3. Target Location
4. Target Description
5. Method of Engagement
6. Method of Control

The observer identification tells the FDC (Fire Direction Center) the identity of the unit that is calling for the artillery rounds.

The warning order sets the parameters of the fire mission, such as the mode (fire for effect, adjust fire, etc.), the size of the fire element (battery, battalion, etc.), and the method used to specify target location (polar, shift from known point, grid, etc.).

The target location is the central, and most difficult aspect, of the fire mission. The techniques for determining target location are varied, from as crude as using the WERM rule (Width = Range X Mils) implemented via an upraised index finger, to as sophisticated as GPS satellites and laser ranging. Without precise location information, even the largest artillery battery will be rendered ineffective. So, whatever method is employed to determine the target's location, it's essential that the forward observer is well trained in the use of that method.

The target description is comprised of the elements that help determine the type and amount of munitions needed. Descriptions should include the type of target, such as "unprotected troops"; the number of elements of the target, such as "three squads"; the degree of protection such as "in tree line"; and finally, if appropriate, the shape of the target, such as "circular, radius 200".

The method of engagement specifies the characteristics of the fire. One such characteristic is the precision of the strike. For nearby targets, where precision is essential to avoid fratricide, the call is "danger close". When this call is given, often the size of area covered by the strike is specified, because the standard 300 meter fire-for-effect sheaf may be too large to employ safely. Another factor is the shell trajectory: The observer might need to call for "high trajectory fire" if the target is defiladed behind

[1] Army Field Manual 6-30

cover. Also, the type of fuze may need to be specified, such as a delayed charge for targets in a tree line, to avoid an ineffective air burst.

The last element of the call for fire is the method of control, which is used to specify the timing of the strike. For example, the call "At my command" means the battery must not fire until the forward observer says to do so, at which time it must fire immediately. Alternatively, the call may be for a strike at a specified clock time. In this case, called "Time On Target", the forward observer tells the FDC a *clock time* for the artillery to fire. This involves an additional step – a "time hack" – to synchronize the clocks of both parties. It may also include a request for illumination rounds interspersed with the actual explosive rounds; and may be either continuous or coordinated (maximum illumination synchronized with explosive impact). If the method of control is "Repeat", another volley is fired using all the same, previously communicated parameters. Conversely, the control message "Check Firing" halts all fires *immediately*, while "Cease Loading" allows loaded guns to fire before ceasing fire.